Acquisition interactive des connaissances d'adaptation intégrée aux sessions de raisonnement à partir de cas — Principes, architecture IakA et prototype KayaK

Amélie Cordier, Béatrice Fuchs, Jean Lieber, Alain Mille

Acquisition interactive des connaissances d'adaptation intégrée aux sessions de raisonnement à partir de cas — Principes, architecture IAKA et prototype KAYAK

Amélie Cordier[1], Béatrice Fuchs[1], Jean Lieber[2] et Alain Mille[1]

[1]LIRIS CNRS, UMR 5202, Université Lyon 1, INSA Lyon, Université Lyon 2, ECL
{Amelie.Cordier, Beatrice.Fuchs, Alain.Mille}@liris.cnrs.fr
[2]Équipe Orpailleur, LORIA UMR 7503 CNRS, INRIA, Universités de Nancy
BP 239 54 506 Vandœuvre-lès-Nancy, France Jean.Lieber@loria.fr

Résumé

En raisonnement à partir de cas, l'acquisition de connaissances au fil des sessions de résolution de problèmes revêt un enjeu particulier ; elle permet l'amélioration progressive des compétences du système sans contrainte pour l'expert. Parmi les differents types de connaissances à acquérir se trouvent les connaissances d'adaptation. L'acquisition opportuniste de connaissances d'adaptation repose sur l'exploitation des interactions entre l'expert et le système. Dans cet article, nous décrivons l'architecture IAKA qui guide la mise en application des principes de l'acquisitions opportuniste. Nous présentons également KAYAK, un prototype développé dans le but de valider l'architecture.

Mots clés : Acquisition interactive de connaissances d'adaptation, réparation, apprentissage par l'échec, méthodes d'adaptation, raisonnement à partir de cas.

1 Introduction

La phase d'adaptation du raisonnement à partir de cas consiste à produire une solution candidate pour résoudre un problème posé. Pour cela, le système utilise un cas remémoré dont il modifie la solution en s'appuyant sur un ensemble de connaissances d'adaptation. La gestion des connaissances d'adaptation dans les systèmes de RÀPC est un problème difficile. En particulier, les connaissances étant souvent locales ou spécifiques à un cas, les représenter demande un important effort d'ingénierie ; il est donc préférable de disposer d'un moyen d'acquérir ces connaissances au fur et à mesure des sessions de résolution de problèmes. Un tel processus peut être qualifié d'acquisition des connaissances *opportuniste* : il s'agit de tirer profit des épisodes de résolution de problèmes en déclenchant un processus d'apprentissage à chaque fois qu'un échec d'adaptation se produit. Dans cet article, nous proposons des principes généraux pour effectuer une acquisition opportuniste des connaissances d'adaptation dans les systèmes de raisonnement à partir de cas. Cette approche repose sur l'exploitation des interactions entre l'expert et le système, en particulier lors de la phase de réparation de la solution. Lorsque l'expert corrige une solution candidate, le système tire profit de ses interactions pour mettre à jour ses connaissances ou en acquérir de nouvelles. L'acquisition des connaissances s'effectue ainsi en collaboration avec l'expert de manière interactive et peu contraignante.

Ce papier est organisé comme suit. Dans la section 2, nous détaillons les principes de l'acquisition opportuniste de connaissances d'adaptation puis, dans la section 3, nous montrons comment l'architecture IAKA guide la mise en application de ces principes. La section 4 présente KAYAK, un prototype développé pour valider l'architecture IAKA et mettant en œuvre l'adaptation différentielle pour résoudre des problèmes d'approximation de fonctions mathématiques complexes. Une expérimentation des principes de IAKA au travers de KAYAK est décrite dans la section 5. La section 6 décrit les travaux connexes et discute de l'approche présentée. Enfin, la section 7 conclut le papier et propose des perspectives de travail pour IAKA et KAYAK.

2 Principes de l'acquisition de connaissances opportuniste

L'acquisition de connaissances opportuniste, aussi appelée acquisition « au fil de l'eau » par les poètes et les lyonnais, consiste à exploiter les actions de l'expert, au moment de la correction d'un cas par exemple, pour apprendre de nouvelles connaissances. Cette approche de l'acquisition des connaissances est motivée par la volonté de réduire l'effort d'ingénierie de la connaissance en sollicitant l'expert de manière ponctuelle lorsqu'il utilise le système pour résoudre un problème. Les stratégies d'acquisition des connaissances « hors ligne »(par exemple, comme dans [7]) c'est-à-dire en dehors de la phase de raisonnement, demandent un effort d'ingénierie important de la part de l'expert. En effet, ce dernier doit, durant une phase de maintenance, traiter un nombre important de connaissances générées par un processus d'apprentissage automatique. Les approches d'acquisition au fil de l'eau et hors ligne peuvent donc être considérées comme complémentaires : alors que l'une permet d'acquérir des connaissances ponctuelles avec un faible effort d'ingénierie des connaissances, l'autre permet de traiter un grand nombre de connaissances mais en demandant un effort plus important.

Dans [3], nous avons établi une typologie des connaissances du raisonnement à partir de cas et nous avons montré en quoi les connaissances de similarité et les connaissances d'adaptation étaient liées. Sur la base de ce constat, nous arguons que les systèmes de RÀPC peuvent tirer un grand profit de l'acquisition opportuniste des connaissances et en particulier des connaissances d'adaptation. Nous avons donc proposé des scénarios d'acquisition de ces connaissances reposant principalement sur le rôle joué par l'expert durant une session de raisonnement. L'un de ces scénarios consiste à exploiter les interactions entre l'expert et le système durant la phase de révision du cycle, c'est-à-dire au moment de la réparation du cas. Le travail présenté dans cet article est une illustration de ce scénario ; ces principes nous ont guidés pour définir l'architecture IAKA décrite dans la section 3.

3 L'architecture IAKA

Cette section décrit les notions et les notations que nous utilisons puis présente l'architecture IAKA ainsi que les principes qui guident l'acquisition de connaissances d'adaptation en exploitant les interactions entre l'expert et le système durant la phase de réparation du cas.

3.1 Notions et notations sur le RÀPC

Le processus du raisonnement à partir de cas peut être décomposé en cinq étapes : élaboration, remémoration, adaptation, test-réparation, mémorisation. L'objectif d'une session de RÀPC est de produire une solution candidate pour résoudre un problème, noté `cible`, construit durant l'étape d'élaboration [12] à partir d'un problème `pré-cible`[1] exprimé par l'utilisateur du système. La construction de la solution de `cible` repose en partie sur la réutilisation d'un cas existant issu de la base de cas. La base de cas contient un ensemble de cas résolus. Un cas est constitué d'une partie problème et d'une partie solution. Durant la phase de remémoration, le système recherche un cas source, noté `cas-srce`, jugé apte à être utilisé pour résoudre `cible`. La sélection de `cas-srce` repose sur un appariement des problèmes sources avec le problème cible. La phase d'adaptation consiste à modifier la solution $\texttt{Sol}(\texttt{srce})$ du cas source remémoré en appliquant les connaissances d'adaptation identifiées par l'appariement. Cette adaptation conduit à l'obtention d'une solution candidate $\widetilde{\texttt{Sol}}(\texttt{cible})$ pour le problème posé. Durant la phase de réparation, l'expert valide ou corrige la solution $\widetilde{\texttt{Sol}}(\texttt{cible})$ proposée par le système. Si la solution candidate est validée, $\widetilde{\texttt{Sol}}(\texttt{cible})$ devient $\texttt{Sol}(\texttt{cible})$: le problème est résolu et un nouveau cas $(\texttt{cible}, \texttt{Sol}(\texttt{cible}))$ est ajouté à la base de cas durant la phase de mémorisation. Dans le cas contraire, le système saisit l'opportunité pour améliorer ses connaissances d'adaptation avant de proposer à l'expert une nouvelle solution adaptée. Ainsi, l'apprentissage a lieu durant les phases de réparation et de mémorisation.

[1] `pré-cible` est une requête de l'utilisateur qui peut être incomplètement exprimée. Le rôle de l'étape d'élaboration est de collecter et d'inférer les informations nécessaires à la construction d'un problème `cible` exploitable par le système.

3.2 Hypothèses sur l'adaptation

Nous faisons l'hypothèse que la phase d'adaptation du cas peut être décomposée en plusieurs étapes. Chaque étape correspond à une opération d'adaptation élémentaire pour résoudre un problème particulier. Une opération d'adaptation élémentaire est réalisée par un opérateur d'adaptation `OA`. L'ensemble des opérations d'adaptation élémentaires permettant de passer de `Sol(srce)` à $\widetilde{\texttt{Sol}(\texttt{cible})}$ est appelé *méthode d'adaptation* noté `MA`. Nous avons donc $\texttt{MA} = \{\texttt{OA}_i\},\ i \in \{1, \ldots n\}$: une méthode d'adaptation est un ensemble de n opérateurs d'adaptation, n étant le nombre d'étapes d'adaptation[2] nécessaires à la résolution de `cible`. À chaque cas est associé un ensemble fini et non vide de méthodes d'adaptation : il peut y avoir une ou plusieurs façons d'adapter un cas source.

3.3 Le cycle du RÀPC re-revisité

L'expression « cycle RÀPC revisité » est utilisée pour caractériser une modification du cycle traditionnel proposé dans [1] intégrant l'étape cruciale[3][4][5][6] qu'est l'élaboration. Quelques années et bien des articles plus tard, nous introduisons une nouvelle version de ce cycle, le cycle re-revisité (cf. figure 2), dans lequel une boucle d'interaction avec l'expert apparaît. Le cycle re-revisité[7] est une adaptation du cycle revisité dans laquelle les interactions entre l'expert et le système sont prises en compte, en particulier lors de l'étape de réparation du cas[8]. On peut noter que ce cycle re-revisité ressemble à une combinaison du cycle revisité et de l'organigramme du RÀPC présenté dans [14].

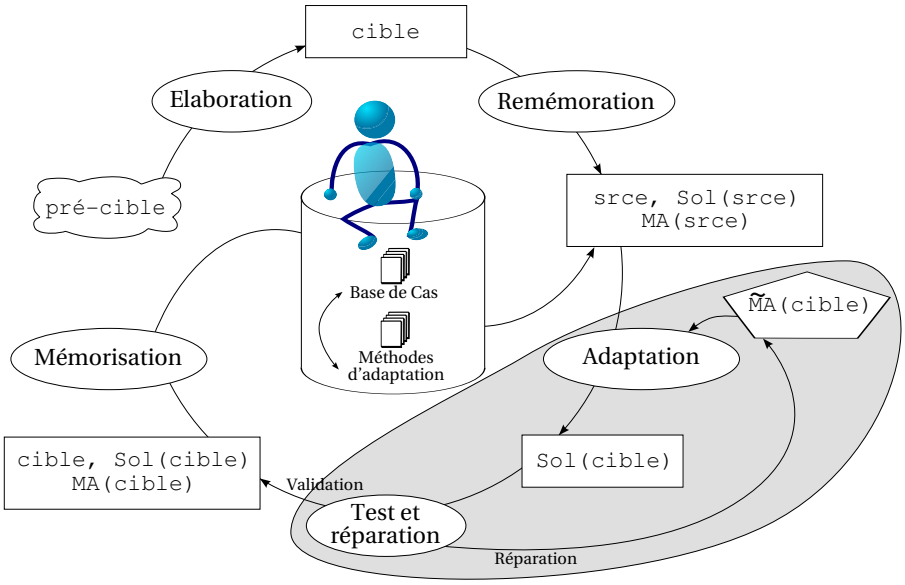

FIG. 1 – Le cycle du RÀPC re-revisité et la boucle d'interaction avec l'expert.

Dans le cycle re-revisité, un problème `cible` est tout d'abord élaboré à partir d'une description de problème `pré-cible` pendant l'étape *d'élaboration*. La phase de *remémoration* est guidée par l'adaptabilité [15], c'est-à-dire qu'elle utilise les connaissances d'adaptation pour pondérer les différences entre `srce` et `cible` afin

[2] n peut très bien être égal à 1, la méthode d'adaptation contient alors un unique opérateur d'adaptation.
[3] En tout cas, c'est l'avis d'Alain Mille.
[4] Amélie Cordier pense qu'effectivement, c'est important.
[5] Béatrice Fuchs est convaincue depuis belle lurette.
[6] Jean Lieber, quant à lui, pense que « cruciale » est un tantinet exagéré.
[7] Notons que, pour la première fois, le cycle contient un, non, deux vrais cycles qui bouclent. Comme quoi, l'intuition de départ était bonne !
[8] Toute ressemblance avec un processus de RÀPC existant serait purement fortuite.

d'évaluer une distance dist qui reflète la difficulté d'adaptation. La distance étant dépendante de la méthode d'adaptation, elle est donc calculée pour chaque couple $(\texttt{srce}, \texttt{MA}(\texttt{srce})_j)$, avec $\texttt{MA}(\texttt{srce})_j$ ($j \in \{1, \dots m\}$) une des méthodes d'adaptation associées à cas-srce. La phase de sélection du cas à utiliser procède donc au choix d'un cas source $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ et d'une méthode d'adaptation associée $\texttt{MA}(\texttt{srce})$. Durant la phase *d'adaptation*, la solution $\texttt{Sol}(\texttt{srce})$ est réutilisée pour construire $\widetilde{\texttt{Sol}}(\texttt{cible})$ en appliquant la méthode d'adaptation $\texttt{MA}(\texttt{srce})$.

$$\mathit{Adaptation} : (\texttt{srce}, \texttt{Sol}(\texttt{srce}), \texttt{cible}, \texttt{MA}(\texttt{srce})) \mapsto \widetilde{\texttt{Sol}}(\texttt{cible})$$

Il faut noter que la méthode d'adaptation prend le problème cible en paramètre. La solution obtenue, $\widetilde{\texttt{Sol}}(\texttt{cible})$ est alors proposée à l'expert pour validation pendant l'étape de *test et réparation*. Deux situations sont alors possibles :

- L'expert juge $\widetilde{\texttt{Sol}}(\texttt{cible})$ satisfaisante : le cas cible résolu $(\texttt{cible}, \texttt{Sol}(\texttt{cible})) = (\texttt{cible}, \widetilde{\texttt{Sol}}(\texttt{cible}))$ est stocké dans la base de cas et $\texttt{MA}(\texttt{cible}) = \texttt{MA}(\texttt{srce})$ est stocké dans la base de méthodes d'adaptation lors de l'étape de *mémorisation*.
- L'expert juge $\widetilde{\texttt{Sol}}(\texttt{cible})$ non satisfaisante, la boucle d'interaction est alors activée. Une méthode d'adaptation $\widetilde{\texttt{MA}}(\texttt{cible})$ est construite à partir d'une copie de $\texttt{MA}(\texttt{srce})$. $\widetilde{\texttt{MA}}(\texttt{cible})$ va ensuite être modifiée par étapes, en fonction des actions de l'expert. Ainsi, les échanges qui ont lieu durant cette boucle d'interaction contribuent à l'acquisition de nouvelles connaissances d'adaptation par la modification de $\widetilde{\texttt{MA}}(\texttt{cible})$ (les détails de la boucle d'interaction sont exposés dans la section 3.4). À chaque modification de $\widetilde{\texttt{MA}}(\texttt{cible})$, le système tente une autre adaptation de $\texttt{Sol}(\texttt{srce})$ pour résoudre cible avec $\widetilde{\texttt{MA}}(\texttt{cible})$ ainsi modifiée, afin de vérifier si la modification effectuée a permis d'obtenir un résultat satisfaisant. Le cycle adaptation-test/réparation se poursuit jusqu'à l'obtention d'une solution acceptée par l'expert.

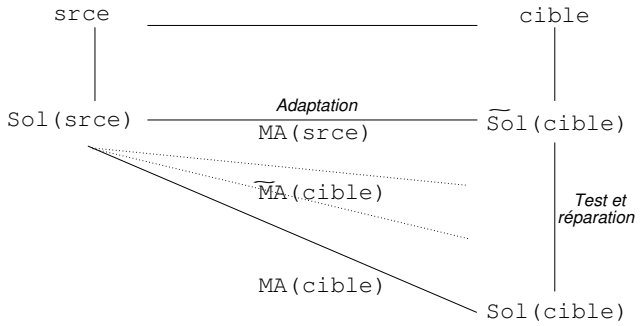

FIG. 2 – Adaptation, test et réparation.

3.4 La boucle d'interaction avec l'expert

Dans cette partie, la boucle d'interaction avec l'expert (cf. figure 3) permettant l'acquisition de connaissances d'adaptation est détaillée.

L'idée générale de l'acquisition opportuniste de connaissances d'adaptation est d'exploiter les corrections apportées à la solution par un expert dans un processus d'apprentissage interactif. Dans IAKA, la boucle d'interaction permet d'améliorer une méthode d'adaptation grâce à la correction des opérateurs d'adaptation la constituant. Lorsqu'une solution a été jugée incorrecte par l'expert, le système cherche à identifier et corriger le ou les opérateurs d'adaptation responsables de l'échec. L'échec peut provenir d'une ou plusieurs étapes constituant l'adaptation, chaque étape, correspondant à l'application d'un opérateur d'adaptation particulier. Il s'agit donc d'identifier parmi les opérateurs d'adaptation de la méthode d'adaptation ceux qui doivent être corrigées. La stratégie de IAKA consiste à tester séparément chacun de ces opérateurs. Pour cela, le système isole un opérateur d'adaptation et l'utilise pour résoudre un nouveau problème pb élaboré spécifiquement à partir de srce afin de le tester en une étape. La solution de ce nouveau problème, obtenue par adaptation, est soumise à l'expert.

La réponse de l'expert permet de décider de la validité de l'opérateur d'adaptation :

– Si l'expert valide l'adaptation effectuée, l'opérateur d'adaptation est considéré comme correct et le système choisit alors un autre opérateur à tester.
– Si l'expert ne valide pas l'adaptation, le système lui demande la solution correcte de pb et modifie l'opérateur d'adaptation en conséquence.

La boucle d'interaction se termine dès qu'un opérateur d'adaptation a été mis à jour ou lorsque tous les opétateurs de la méthode ont été étudiés par l'expert. La figure 3 résume ce processus.

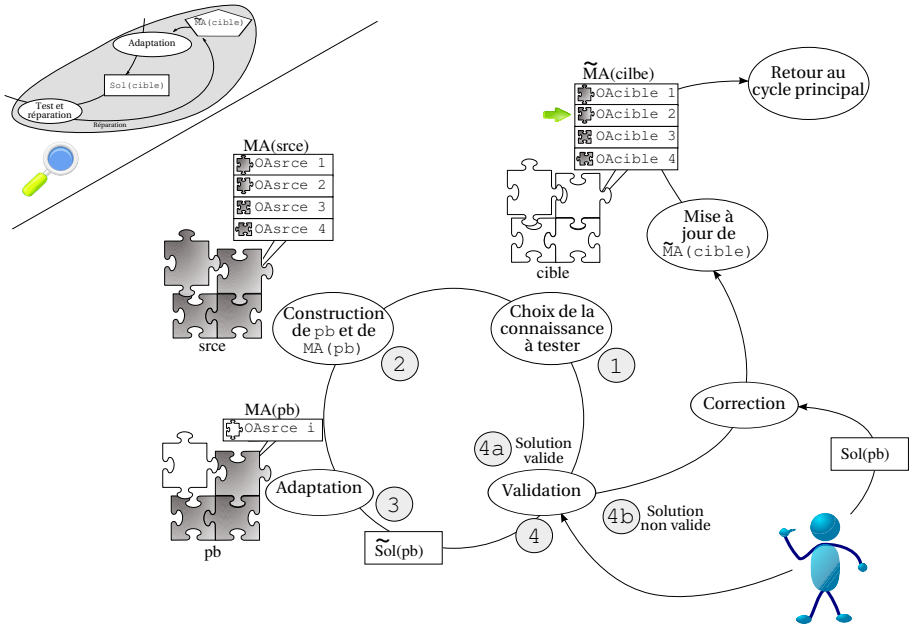

FIG. 3 – La boucle d'interaction avec l'expert.

1. Sélection d'un opérateur d'adaptation $\texttt{OA(cible)}_\texttt{i}$ à tester parmi les opérateurs de $\widetilde{\texttt{MA}}\texttt{(cible)}$ qui n'ont pas encore été testés.
2. Construction de pb à partir de srce en remplaçant une partie de srce par la partie de cible justifiant l'application de $\texttt{OA(cible)}_\texttt{i}$. La méthode d'adaptation associée MA(pb) est constituée du seul opérateur d'adaptation $\texttt{OA(cible)}_\texttt{i}$. On a donc : $\texttt{MA(pb)} = \{\texttt{OA(cible)}_\texttt{i}\}$.
3. L'adaptation de (srce, Sol(srce)) avec la méthode MA(pb) pour résoudre pb donne $\widetilde{\texttt{Sol}}\texttt{(pb)}$.
4. $\widetilde{\texttt{Sol}}\texttt{(pb)}$ est présentée à l'expert qui valide ou non cette solution. Deux situations sont alors possibles :
   (a) L'expert valide $\widetilde{\texttt{Sol}}\texttt{(pb)}$. Un autre opérateur à tester sera choisi lors du retour à l'étape 1.
   (b) L'expert ne valide pas $\widetilde{\texttt{Sol}}\texttt{(pb)}$.
      i. Le système lui demande la valeur de Sol(pb) et corrige MA(pb) en modifiant $\texttt{OA(cible)}_\texttt{i}$ (la seule que l'on puisse remettre en cause).
      ii. $\widetilde{\texttt{MA}}\texttt{(cible)}$ est mise à jour en remplaçant l'ancien $\texttt{OA(cible)}_\texttt{i}$ par le $\texttt{OA(cible)}_\texttt{i}$ nouvellement corrigé et une nouvelle adaptation est effectuée pour tester l'impact de l'opérateur d'adaptation modifié.

4 KAYAK : un prototype pour l'architecture IAKA

KAYAK est un prototype d'application de RÀPC développé pour implanter et valider les principes de l'architecture IAKA.

4.1 Domaine d'application

Le domaine d'application de KAYAK est celui des fonctions à n variables. Dans ce prototype, nous avons choisi la fonction f définie par :

$$f(x_1, x_2, x_3) = \frac{\sin x_1 . e^{x_2}}{1 + |x_2 + x_3|}$$

Résoudre un problème dans ce prototype consiste à déterminer une approximation de la valeur de $f(x_1, x_2, x_3)$ pour x_1, x_2 et x_3 donnés. L'expert du domaine est simulé par un couple (f, ε) où f est la fonction mathématique (que l'on cherche à approcher) et ε (avec $\varepsilon > 0$) est un paramètre représentant le degré d'exigence de l'expert.

Dans une session de raisonnement de KAYAK, les solutions proposées par le système sont validées et corrigées par l'expert. Si l'expert juge une solution non satisfaisante, un processus de correction est mis en œuvre et des opérateurs d'adaptation sont identifiés et modifiés. La fonction f, particulièrement serviable, nous aide également à constituer notre base de cas et à définir les opérateurs d'adaptation qui prennent la forme de fonctions d'influence.

La base de cas initiale est construite en utilisant la fonction f. Un ensemble de triplets est généré et, pour chaque triplet, la valeur de f correspondante est calculée. Le triplet (partie problème) et la valeur de f correspondante (partie solution) constituent un cas qui est stocké dans une base de cas. Un cas est défini ainsi :

$$\begin{aligned} \texttt{srce} &= \{x_1, x_2, x_3\} \\ \texttt{cible} &= y \end{aligned}$$

Lors de la construction d'un cas, une méthode d'adaptation associée est calculée et ajoutée à la base de méthodes d'adaptation. Un lien est établi entre le cas et sa méthode d'adaptation définie par l'expert. Dans la version courante de KAYAK, la base de cas est générée par un processus aléatoire (qui s'appuie sur une distribution uniforme de probabilités définie sur le domaine de définition de f), mais une approche différente peut être envisagée (discutée dans la section 7).

4.2 Remémoration et adaptation

Le prototype KAYAK repose sur l'architecture de IAKA où le processus adaptation est réalisé grâce à la stratégie de l'adaptation différentielle [8].

Adaptation différentielle. Dans l'adaptation différentielle, les écarts constatés entre un problème `srce` et un problème `cible` sont exploités par les *dépendances*. Une dépendance indique un lien entre un descripteur de problème et un descripteur de solution. À ce lien est associée une *fonction d'influence* qui exprime la variation d'un descripteur de solution induite par la variation d'un descripteur de problème. Adapter un cas en utilisant l'adaptation différentielle consiste donc à modifier `Sol(srce)` en utilisant les influences pertinentes. Ainsi, dans la stratégie d'adaptation différentielle, la variation d'un descripteur de solution donné y_j est obtenue en combinant les fonctions d'influence $\frac{\partial y_j}{\partial x_i}$ et les variations des descripteurs de problème. Par analogie avec les dérivées partielles, on obtient :

$$dy_j = \sum_i \frac{\partial y_j}{\partial x_i} \times dx_i$$

Adaptation dans KAYAK. L'objectif de l'étape d'adaptation dans ce système est d'estimer la valeur y^c de la fonction f étant donné les valeurs des variables, notées x_i^c, par adaptation de la solution $\texttt{Sol}(\texttt{srce}) = y^s$ d'un cas `cas-srce` remémoré. La méthode d'adaptation $\texttt{MA}(\texttt{srce})$ associée à `cas-srce` contient un ensemble de dépendances à appliquer[9]. Les dépendances jouent le rôle d'opérateurs d'adaptation. Chacune des dépendances est constituée d'une fonction d'influence indiquant comment varie un descripteur de solution en fonction d'un descripteur de problème. Dans ce domaine, les fonctions d'influence sont les dérivées partielles de la fonction f par rapport à chacun des descripteurs de problèmes ; la valeur de la dérivée partielle en un point est le coefficient de variation. La solution y^c de `cible` est calculée en appliquant l'ensemble des dépendances. Pour chaque dépendance, l'écart entre le descripteur de problème x_i^s de `srce` et x_i^c de `cible` est calculé, il est ensuite multiplié par le coefficient de variation puis ajouté à la valeur du descripteur de solution source y^s pour donner la valeur du descripteur de solution cible y^c. Pour cette application, on a donc :

$$\widetilde{y^c} = y^s + \sum_{i=1}^{3} \frac{\partial y}{\partial x_i} \times dx_i$$

La valeur d'une influence $\frac{\partial y_j}{\partial x_i}$ peut être calculée de deux manières : soit en utilisant l'expression générale de la dérivée partielle de f par rapport à x_i ($\frac{\partial f}{\partial x_i}$), soit en calculant une valeur approchée du taux de variation sur un petit intervalle. La première méthode a l'avantage de donner la valeur exacte de la dérivée de f en un point tandis que la deuxième permet d'éviter d'avoir recours à la connaissance de la forme analytique de la dérivée partielle de f. C'est la deuxième approche qui est utilisée dans KAYAK.

Remémoration. Dans un contexte de remémoration guidée par l'adaptabilité [15], la distance prend en compte la méthode d'adaptation associée au cas. Lorsque plusieurs méthodes d'adaptation sont associées à un cas, il y a autant de distances calculées que de méthodes d'adaptation disponibles pour ce cas. Nous faisons l'hypothèse que l'effort d'adaptation peut être mesuré par une distance `dist` de $\texttt{Sol}(\texttt{srce})$ à $\texttt{Sol}(\texttt{cible})$, notée $\texttt{dist}(\texttt{Sol}(\texttt{srce}), \texttt{Sol}(\texttt{cible}))$. On peut montrer que cette distance peut s'exprimer comme une distance $\texttt{dist}(\texttt{srce}, \texttt{cible})$ entre les parties problèmes étant données les connaissances d'adaptation. La formule générale de la distance utilisée dans l'algorithme suivant s'exprime donc par :

$$\texttt{dist}(\texttt{srce}, \texttt{cible}) = \sum_i w_i (x_i^c - x_i^s)^2$$

$$w_i = \sum_j (\frac{\partial y_j}{\partial x_i})^2$$

Conformément à l'architecture IAKA, la remémoration retourne un couple $(\texttt{cas-srce}, \texttt{MA}(\texttt{srce}))$.

4.3 Interactions avec l'expert virtuel

On distingue deux modes de consultation de l'expert : validation et correction de la solution.

Pour juger de la validité d'une solution, l'expert s'appuie sur le paramètre ε. Afin de déterminer si $\widetilde{\texttt{Sol}}(\texttt{cible})$ est valide, l'expert calcule l'écart entre $\widetilde{\texttt{Sol}}(\texttt{cible})$ et $\texttt{Sol}(\texttt{cible})$ et le compare à ε. Ainsi, si $|\widetilde{\texttt{Sol}}(\texttt{cible}) - \texttt{Sol}(\texttt{cible})| \leq \varepsilon$, $\widetilde{\texttt{Sol}}(\texttt{cible})$ est acceptée par l'expert. Le seuil de tolérance est modifié lorsque l'expert doit juger de la validité d'une solution $\widetilde{\texttt{Sol}}(\texttt{pb})$ pour un problème `pb` qui a été construit pour tester un opérateur d'adaptation particulier. Dans ce cas, ce seuil passe à $\frac{\varepsilon}{n}$, n étant le nombre d'opérateurs d'adaptation de la méthode. En effet, nous faisons l'hypothèse que $\widetilde{\texttt{Sol}}(\texttt{cible})$ résulte de l'application de n opérateurs d'adaptation. Par conséquent, si $\widetilde{\texttt{Sol}}(\texttt{cible})$ correspond à une erreur supérieure à ε, c'est que l'un au moins des opérateurs d'adaptation provoque une erreur supérieure à $\frac{\varepsilon}{n}$.

Pour corriger une solution $\widetilde{\texttt{Sol}}(\texttt{pb})$, l'expert calcule simplement la solution de `pb` en appliquant f et renvoie le résultat.

[9] Nous nous limitons ici aux dépendances numériques simples faisant intervenir des réels.

Même si, dans KAYAK, ces deux modes de consultation demandent le même effort de calcul (il faut évaluer f à chaque fois), nous les considérons comme deux opérations différentes et les comptabilisons séparément.

4.4 Algorithme de KAYAK

Entrées : Base de cas (`BaseDeCas`), base de méthodes d'adaptation (`BaseDeMA`),
expert (f, ε), un problème cible (`cible`).
Compteurs : Les compteurs sont utilisés pour l'évaluation.
$Compteur_{test}$: nombre de consultations de l'expert pour validation.
$Compteur_{reparation}$: nombre de consultations de l'expert pour la correction d'une solution.

début (algorithme)
  / Remémoration */*
  (`srce`, `Sol`(`srce`)) et `MA`(`srce`) ← `remémoration` avec (`BaseDeCas`, `cible`) selon `dist`
  / Adaptation */*
  $\widetilde{\texttt{Sol}}$(`cible`) ← `AdaptationDifferentielle`(`srce`, `Sol`(`srce`), `cible`, `MA`(`srce`))
  / Test et réparation */*
  si non validation de l'expert pour (`cible`, $\widetilde{\texttt{Sol}}$(`cible`)) */* tolérance : ε */*
  alors
    $\widetilde{\texttt{MA}}$(`cible`) ← `MA`(`srce`)
    Répeter
    / Test */*
    `OK` ← validation de l'expert pour (`cible`, $\widetilde{\texttt{Sol}}$(`cible`)) */* tolérance : ε */*
    $Compteur_{test} \leftarrow Compteur_{test} + 1$
    si Non `OK`
    alors
      tant que Non `OKpb` ou `OA`(`cible`)$_{\texttt{i}}$ à tester
        `OA`(`cible`)$_{\texttt{i}}$ ← `ChoixAleatoire`($\widetilde{\texttt{MA}}$(`cible`))
        `pb` ← `ConstructionPb`(`srce`, `cible`)
        `MA`(`pb`) ← {`OA`(`cible`)$_{\texttt{i}}$}
        $\widetilde{\texttt{Sol}}$(`pb`) ← `AdaptationDifferentielle`(`srce`, `Sol`(`srce`), `pb`, `MA`(`pb`))
        `OKpb` ←validation de l'expert pour (`pb`, $\widetilde{\texttt{Sol}}$(`pb`)) */* tolérance : $\frac{\varepsilon}{n}$ */*
        $Compteur_{test} \leftarrow Compteur_{test} + 1$
      fin (tant que)
      si Non `OKpb`
      / Réparation */*
      alors
        `Sol`(`pb`) ←réparation de l'expert pour (`pb`, $\widetilde{\texttt{Sol}}$(`pb`))
        $Compteur_{reparation} \leftarrow Compteur_{reparation} + 1$
        $\widetilde{\texttt{MA}}$(`cible`) ← `RéparationMAcible`($\widetilde{\texttt{MA}}$(`cible`), `OA`(`cible`)$_{\texttt{i}}$, `pb`, `Sol`(`pb`))
      fin (si)
    fin (si)
    $\widetilde{\texttt{Sol}}$(`cible`) ← `AdaptationDifferentielle`(`srce`, `Sol`(`srce`), `cible`, $\widetilde{\texttt{MA}}$(`cible`))
    Jusqu'à OK ou `OA`(`cible`)$_{\texttt{i}}$ tous testés
  fin (si)
  `MA`(`cible`) ← $\widetilde{\texttt{MA}}$(`cible`)
  `Mémoriser`(`cible`, `Sol`(`cible`)) dans `BaseDeCas`, et `MA`(`cible`) dans `BaseDeMA`
fin (algorithme)

Algorithme de KAYAK.

ChoixAleatoire. Cette méthode consiste à choisir aléatoirement dans l'ensemble des opérateurs d'adaptation contenus dans la méthode d'adaptation un opérateur à tester. Les opérateurs déjà testés sont marqués comme tels et ne peuvent pas être choisis à nouveau. D'autre part, on ne traite que les opérateurs pour lesquels il existe un écart entre les descripteurs de problème qui justifient leur application. Comme il n'existe pas de moyen de savoir quels opérateurs d'adaptation doivent être modifiés, il n'est pas possible de mettre en place une stratégie pour traiter prioritairement les « mauvais » opérateurs. Par conséquent, nous sélectionnons les opérateurs à traiter aléatoirement afin de ne pas renforcer l'amélioration d'un opérateur au détriment des autres, ce qui serait le cas si le système les traitait toujours dans le même ordre.

ConstructionPb. La construction de `pb` est effectuée dès que l'opérateur d'adaptation à tester a été choisi. Le problème `pb` est construit de sorte à pouvoir tester l'opérateur $\texttt{OA(cible)}_\texttt{i}$ en une seule étape. Ce problème est donc construit à partir du problème `srce` dans lequel on remplace une partie par la partie correspondante de `cible`. La partie à remplacer est choisie en fonction de $\texttt{OA(cible)}_\texttt{i}$: c'est une partie qui diffère entre `srce` et `cible` et qui justifie l'application de l'opérateur d'adaptation. La méthode d'adaptation permettant de calculer $\texttt{Sol(pb)}$ est construite en même temps et contient le seul opérateur d'adaptation $\texttt{OA(cible)}_\texttt{i}$.

RéparationMAcible. Lorsque l'expert corrige la solution d'un problème `pb`, le système est en mesure d'acquérir une nouvelle connaissance d'adaptation. En effet, il lui est possible de corriger l'opérateur d'adaptation $\texttt{OA(cible)}_\texttt{i}$ qui était testé par `pb`. Dans cette application, $\texttt{OA(cible)}_\texttt{i}$ met en relation x_i et y au travers d'un coefficient noté $coef$[10] qui permet de calculer la variation de y en fonction de la variation de x_i^s. Corriger $\texttt{OA(cible)}_\texttt{i}$ revient donc à corriger le coefficient $coef$, c'est-à-dire à donner une meilleure approximation de la pente de f en x_i^c, avec $coef = f'(x_i^s)$. On peut observer (cf. figure 4) que la pente de la corde reliant les points $(x_i^s, f(x_i^s))$ et $(x_i^c, f(x_i^c))$ est une moyenne des pentes de f en x_i^s et en x_i^c.

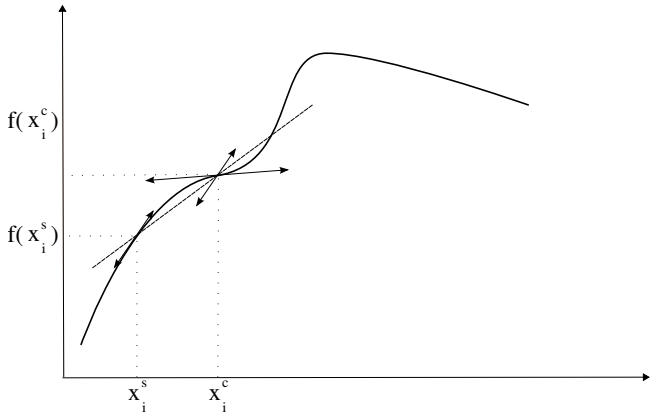

FIG. 4 – Correction de $\texttt{OA(cible)}_\texttt{i}$.

Ainsi, on peut approximer la pente de f en x_i^c par :

$$f'(x_i^c) = 2 \times \frac{f(x_i^c) - f(x_i^s)}{x_i^c - x_i^s} - f'(x_i^s)$$

On peut montrer que cette approximation est assez bonne dans tous les cas.

Une fois l'opérateur d'adaptation corrigé, la méthode d'adaptation $\widetilde{\texttt{MA}}\texttt{(cible)}$ est mise à jour et une nouvelle adaptation pour essayer de calculer $\texttt{Sol(cible)}$ est effectuée.

[10] $coef$ correspond à la pente de f au point x_i

5 Perspectives d'expérimentations

Dans la version actuelle de KAYAK, le cycle complet du RÀPC et la boucle d'interaction avec l'expert (virtuel) sont complètement implantés. Le processus de génération aléatoire de la base de cas et l'expert sont également implantés et fonctionnels. Ainsi, le système est en mesure d'effectuer des sessions complètes de résolution de problème : élaboration par génération aléatoire d'un problème cible, remémoration selon la distance `dist` définie dans la section 4.2, bouclage adaptation-test/réparation qui s'effectue jusqu'à l'obtention d'une solution acceptable pour le problème cible et enfin, mémorisation du résultat de la session de résolution de problème. L'acquisition des connaissances d'adaptation est préparée durant la phase de test-réparation puis se concrétise durant la mémorisation.

L'étape suivante dans le développement de KAYAK consiste à mettre en place des tests pour valider l'approche et évaluer le prototype. Les trois sections suivantes sont consacrées à la présentation des tests et des évaluations que nous envisageons d'effectuer dans un futur proche.

5.1 Génération de la base de cas

La génération de la base de cas mise en place dans KAYAK consiste à choisir les valeurs des trois descripteurs x_1, x_2, x_3 de façon aléatoire et indépendante puis de calculer grâce à f la solution y exacte correspondante ainsi que la méthode d'adaptation par défaut associée. Cette approche ne garantit pas une répartition uniforme des cas dans la base de cas initiale. Afin de pallier ce problème, nous proposons un algorithme de génération d'une base de cas initiale fondé sur la notion de « rejet » qui assure une meilleure couverture du domaine d'application par la base de cas initiale. Dans l'agorithme présenté ci-dessous, on utilise la notion de $\texttt{dist}(\texttt{BaseDeCas}, \texttt{cible})$ qui donne la distance d'un problème cible à la base de cas. Cette distance est définie ainsi :

$$\texttt{dist}(\texttt{BaseDeCas}, \texttt{cible}) = min(\texttt{dist}(\texttt{srce}, \texttt{cible}) | (\texttt{srce} \in \texttt{BaseDeCas}))$$

L'algorithme utilise une variable K qui est utilisée pour définir la limite en deça de laquelle un cas est considéré comme « trop proche » et donc rejeté. La valeur de K évolue au fur et à mesure que l'on remplit la base de cas.

Cet algorithme permet donc de générer une base de cas initiale ayant une bonne « couverture du domaine » : il y a plus de cas dans les zones de l'espace de problème dans lesquelles les variations sont importantes[11].

Implanter ce second algorithme dans KAYAK permettrait d'observer le rôle joué par la couverture de la base de cas dans la compétence du système. En résolvant une même série de problèmes avec une base de cas aléatoire d'une part et avec une base de cas unifome d'autre part, on pourrait comparer l'impact de la couverture de la base de cas initiale sur l'efficacité du système. On peut supposer qu'un système résoud, en moyenne, plus rapidement un cas lorsque la couverture de la base de cas initiale est bonne. La problématique sous-jacente évoquée ici est celle de l'acquisition initiale des connaissance dans un système à base de connaissances.

5.2 Validation de la mesure de similarité

KAYAK est un bon terrain pour expérimenter l'impact de la remémoration sur la qualité de l'adaptation. Dans ce prototype, nous utilisons les connaissances d'adaptation pour pondérer la remémoration. Une des expérimentations que nous envisageons consiste à effectuer des sessions de résolution de problème en utilisant une remémoration simple d'une part (reposant sur la comparaisons des écarts entre les descripteurs de problèmes) et en utilisant la remémoration guidée par l'adaptabilité d'autre part. Les expérimentations pourraient permettre d'évaluer :

- l'intérêt d'une bonne remémoration sur la qualité de l'adaptation (remémoration guidée par l'adaptabilité)
- l'impact, au niveau de l'étape de remémoration, de l'utilisation des connaissances d'adaptation acquises interactivement.

On pourrait également envisager de montrer par l'exemple l'assertion $\texttt{dist}(\texttt{Sol}(\texttt{srce}), \texttt{Sol}(\texttt{cible})) = \texttt{dist}(\texttt{srce}, \texttt{cible})$ présentée dans la section 4.2.

[11] Remarque :la génération des méthodes d'adaptation associées à chaque cas ne figure pas dans l'algorithme.

Paramètres :
$nbCas$ (nombre de cas à générer),
$FacDec$ (facteur de correction de K)

début (algorithme)
  $\texttt{BaseDeCas} \leftarrow \{\}$, $K \leftarrow +\infty$, $i \leftarrow 0$
  tant que $i \leq nbCas$
    $\texttt{pb} \leftarrow$ GenénérerPbAléatoirement()
    si $dist(\texttt{BaseDeCas}, \texttt{pb}) \leq (k/nbCas)$
    alors
      Calculer($\texttt{Sol}(\texttt{pb})$, $\frac{\partial y_j}{\partial x_i}, i \in \{1, \ldots n\}$)
      si $i = 0$
      alors
        $K \leftarrow Calcul()$
      fin (si)
      $\texttt{BaseDeCas} \leftarrow \texttt{BaseDeCas} \cup (\texttt{pb}, \texttt{Sol}(\texttt{pb}))$
      $i \leftarrow i + 1$
    sinon
      $K \leftarrow K \times FacDec$
    fin (si)
  fin (tant que)
fin (algorithme)

Algorithme de génération de la base de cas basé sur le rejet.

5.3 Qualité et utilité des connaissances acquises

L'évaluation de la qualité et de l'utilité des connaissances d'adaptation acquises constitue la priorité principale dans le plan d'évaluation de KAYAK. Nous souhaitons montrer que les connaissances acquises sont de bonne qualité, qu'elles sont utilisables et qu'elles améliorent la compétence du système. L'évaluation de la qualité des connaissances apprises peut se faire à la fois au niveau des opérateurs d'adaptation et au niveau des méthodes d'adaptation elles-mêmes.

Le premier test consiste à résoudre une série de problèmes et à observer au bout de combien d'essais le système parvient à résoudre un certain nombre de problèmes sans se tromper. Dans un deuxième temps, pour tester l'utilité des connaissances apprises, il sera nécessaire de faire des test comparatifs en résolvant une série de problèmes en mémorisant les connaissances d'adaptation acquises et la même série de problème sans utiliser ces connaissances. Il est également envisagé de faire des tests locaux en utilisant la stratégie *leave-one-out*, c'est-à-dire en résolvant un cas une première fois, sans mémoriser sa solution puis en essayant de le résoudre à nouveau afin de vérifier si les connaissances acquises dans un premier temps sont exploitées.

Nous nous interrogeons également sur l'utilité de mémoriser les problèmes intermédiaires générés afin de tester les différents opérateurs d'adaptation. Des tests empiriques permettront de mettre en évidence les avantages et les inconvénients d'une telle stratégie.

6 Discussion et travaux proches

Le RÀPC permet de proposer des hypothèses de solution à de nouveaux problèmes, même en présence d'une théorie faible ou incomplète. Néanmoins, faute de connaissances suffisantes, ces solutions peuvent ne pas convenir ce qui conduit à un échec de raisonnement. Cet échec est à l'origine d'un processus de réparation permettant d'apprendre de nouvelles connaissances. De nombreux travaux se sont intéressés à la composante apprentissage du RÀPC qui revêt plusieurs aspects.

Le premier aspect concerne les conteneurs de connaissances ciblés pour l'apprentissage [13] : les cas, les connaissances de similarité, les connaissances d'adaptation, les connaissances du domaine. Certaines approches, comme [5] par exemple, considèrent séparément les connaissances de similarité et les connaissances d'adaptation ainsi que leur apprentissage. Pour IAKA au contraire, la connaissance essentielle qu'il est pertinent d'apprendre est la connaissance d'adaptation car c'est elle qui pilote toutes les étapes du cycle du RÀPC.

Un deuxième aspect est lié au mode de déclenchement de l'apprentissage . L'apprentissage «au fil de l'eau» ou opportuniste tel que réalisé dans IAKA consiste à tirer profit de chaque épisode de résolution de problème pour détecter des connaissances erronées et lancer un processus d'apprentissage en cas de besoin. L'apprentissage est «incrémental» et quelques connaissances «locales» sont remises en cause et éventuellement améliorées. Parmi les systèmes de cette catégorie, on peut citer [11; 9]. Dans les approches «hors ligne» au contraire, l'apprentissage de connaissances est réalisé en dehors du cycle du RÀPC. L'apprentissage a lieu en général dans le cadre d'un processus distinct du RÀPC tel que par exemple un processus d'ECBD ou d'apprentissage automatique [10; 4; 6].

Enfin un dernier aspect concerne la source de connaissances utilisée pour l'apprentissage [16]. Certaines approches utilisent uniquement les connaissances présentes dans les conteneurs, notamment celles qui s'appuient sur l'apprentissage automatique.D'autres au contraire visent à acquérir des connaissances qui ne sont pas déjà présentes dans le système ou à améliorer les connaissances existantes par interaction avec l'environnement [3; 11; 2]. IAKA relève de cette dernière catégorie. L'apprentissage se déroule pendant l'utilisation du système et vise à raffiner les connaissances d'adaptation existantes.

La difficulté principale de l'apprentissage opportuniste est de mettre en place des interactions avec l'utilisateur de la façon la moins contraignante possible tout en essayant d'obtenir suffisamment d'informations pour apprendre des connaissances. L'approche IAKA présente l'avantage d'être simple pour l'expert qui est sollicité de 2 façons : soit pour donner son accord ou son désaccord sur la solution proposée par le système, soit pour donner des valeurs de solutions. Dans d'autres cas plus complexes, l'expert du domaine ne pourra que partiellement fournir les informations et il devra être assisté d'un informaticien pour maintenir la base de connaissances en accord avec ses connaissances.

Nous avons testé cette approche dans un domaine qui peut être modélisé à l'aide d'opérateurs d'adaptation différentielle sur des caractéristiques numériques. Il faut encore étudier d'autres domaines avec des caractéristiques symboliques et prendre en compte les connaissances du domaine qui peuvent contraindre l'applicabilité des opérateurs d'adaptation. Néanmoins l'algorithme de réparation présenté est relativement générique car il suffit d'adapter la procédure de choix de la connaissance à tester dans la boucle de réparation qui peut se faire à l'aide de connaissances stratégiques et de contraintes du domaine. De la même façon, les procédures de modification d'un opérateur d'adaptation en fonction du retour de l'expert peut être adaptée en fonction du domaine d'application.

Nous n'avons testé dans IAKA que le cas de figure où l'échec de la solution provient d'un opérateur d'adaptation. Il reste encore à étudier l'éventualité que des connaissances soient manquantes dans la base de connaissances du domaine, comme cela est le cas dans le système FRAKAS [2]. Une interaction plus élaborée avec l'utilisateur serait alors nécessaire afin de diagnostiquer la cause de l'échec et de déterminer la réparation nécessaire.

7 Conclusion et perspectives

Dans cet article, nous avons présenté les principes de l'acquisition opportuniste des connaissances d'adaptation en raisonnement à partir de cas puis nous avons décrit l'architecture IAKA qui guide le développement d'applications de RÀPC qui reposent sur ces principes. Nous avons proposé une formalisation de la boucle d'interaction avec l'expert qui permet l'acquisition interactive des connaissances d'adaptation au travers d'opérateurs et de méthodes d'adaptation. Enfin, nous avons présenté KAYAK, un prototype développé dans le but de valider les idées de IAKA.

KAYAK est actuellement capable de résoudre des problèmes d'approximation de fonctions en appliquant le raisonnement à partir de cas. Il implante un cycle complet de raisonnement intégrant la boucle d'interaction avec l'expert et les différentes phases de l'acquisition des connaissances d'adaptation. Néanmoins, un important

travail d'évaluation reste à effectuer. Les perspectives immédiates de ce travail sont donc de mettre en place les protocoles d'évaluation que nous avons déjà défini pour valider nos hypothèses principales.

Le prototype KAYAK met en œuvre la notion de confiance qui permet d'estimer la qualité des cas et des connaissances d'adaptation contenues dans le système. Les cas et les connaissances de la base initiale sont considérées comme certaines. Les cas résolus et les connaissances acquises ont un degré de certitude associé moindre. La confiance associée à un élément est attribué de manière dégressive : si une connaissance c_j est obtenue à partir d'une connaissance c_i, le degré de confiance de c_j sera inférieur à celui de c_i. La gestion de la confiance est effectuée de manière automatique. Une perspective importante est d'approfondir cette notion de confiance en particulier pour permettre à l'utilisateur d'ajuster lui même les seuils pour les cas et les connaissances.

Nous souhaitons également utiliser KAYAK comme base de travail pour le développement d'un prototype plus évolué dans lequel l'expert ne serait plus un expert virtuel docile mais un véritable expert, d'un véritable domaine.

Enfin, dans notre prototype, l'adaptation est réalisée par la stratégie d'adaptation différentielle numérique, or, l'architecture IAKA n'impose aucune restriction particulière sur la stratégie d'adaptation à utiliser. Nous souhaitons expérimenter l'implantation d'autres stratégies d'adaptation dans le prototype ; cela implique d'une part l'extension des opérateurs existants (qui pour l'instant ne permettent de manipuler que les réels) et d'autre part, la mise en place de nouveaux opérateurs correspondant à de nouvelles stratégies d'adaptation. L'adaptation différentielle se montre particulièrement efficace pour traiter la plupart des descripteurs de type numérique. On peut montrer qu'il est possible de faire le même type d'adaptation sur des descripteurs symboliques, même si cela est moins intuitif. Lorsque les descripteurs sont symboliques, il est souvent plus simple d'avoir recours à d'autres stratégies d'adaptation comme l'adaptation conservatrice [2] par exemple[12]. Une réflexion est en cours sur la façon dont il est possible d'intégrer l'adaptation différentielle et l'adaptation conservatrice au sein d'un architecture comme IAKA. Est-il possible d'unifier les deux stratégies ? L'une des deux doit-elle piloter l'autre ? Afin d'apporter des élements de réponses, nous voulons faire évoluer KAYAK dans le but d'intégrer les deux stratégies et de montrer les apports de chacune : y'a plus qu'a !

Remerciements

Nous tenons à remercier notre experte, (f, ε), pour le temps qu'elle nous a consacré et pour la rigueur de son travail ainsi que la qualité de ses résultats.

Références

[1] A. Aamodt et E. Plaza. Case-based Reasoning : Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1) :39–59, 1994.

[2] A. Cordier, B. Fuchs, J. Lieber, et A. Mille. Failure analysis for domain knowledge acquisition in a knowledge-intensive cbr system. In Rosina Weber Michael Richter, editor, *Proceedings of the 7th international conference on case-based reasoning (to appear)*, LNAI. Springer, 2007.

[3] A. Cordier, B. Fuchs, et A. Mille. Engineering and Learning of Adaptation Knowledge in Case-Based Reasoning. In *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW-06)*, pages 303–317, 2006.

[4] S. Craw, N. Wiratunga, et R. C. Rowe. Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170(16–17) :1175–1192, 2006.

[5] Susan Craw. Introspective learning to build case-based reasoning (cbr) knowledge containers. In *Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 03)*, LNAI 2734, pages pages 1–6, Leipzig, Germany, 2003. Springer.

[12] On ne peut pas s'empêcher de faire le rapprochement avec FRAKAS

[6] M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, et L. Szathmary. Case Base Mining for Adaptation Knowledge Acquisition. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 750–755, 2007.

[7] Mathieu D'Aquin, Fadi Badra, S. Lafrogne, Jean Lieber, Amedeo Napoli, et L. Szathmary. Case Base Mining for Adaptation Knowledge Acquisition. In *Twentieth International Joint Conference on Artificial Intelligence - IJCAI'07*, pages 750–755, 2007.

[8] Béatrice Fuchs, Jean Lieber, Alain Mille, et Amedeo Napoli. A general strategy for adaptation in case-based reasoning. Technical Report RR-LIRIS-2006-016, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon, 2006.

[9] K. J. Hammond. Explaining and Repairing Plans That Fail. 45(1–2) :173–228, 1990.

[10] K. Hanney et M. T. Keane. Learning Adaptation Rules From a Case-Base. In I. Smith et B. Faltings, editors, *Advances in Case-Based Reasoning – Third European Workshop, EWCBR'96*, LNAI 1168, pages 179–192. Springer Verlag, Berlin, 1996.

[11] D. B. Leake, A. Kinley, et D. C. Wilson. Learning to Integrate Multiple Knowledge Sources for Case-Based Reasoning. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 246–251, 1997.

[12] Jean Lieber, Béatrice Fuchs, Alain Mille, et Amedeo Napoli. Une première formalisation de la phase d'élaboration du raisonnement à partir de cas. In *14ème atelier francophone de raisonnement à partir de cas*, 2006.

[13] M. M. Richter. The Knowledge Contained in Similarity Measures. Invited Talk of the First International Conference on Case-Based Reasoning, (ICCBR'95), 1995.

[14] C. K. Riesbeck et R. C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[15] B. Smyth et M. T. Keane. Retrieving Adaptable Cases. In S. Wess, K.-D. Althoff, et M.M. Richter, editors, *Topics in Case-Based Reasoning – First European Workshop (EWCBR'93), Kaiserslautern*, LNAI 837, pages 209–220. Springer, Berlin, 1994.

[16] W. Wilke, I. Vollrath, K.-D. Althoff, et R. Bergmann. A Framework for Learning Adaptation Knowedge Based on Knowledge Light Approaches. In *Proceedings of the Fifth German Workshop on Case-Based Reasoning*, pages 235–242, 1997.